The relationship between CC and SLOC : a preliminary analysis on its evolution

The relationship between CC and SLOC: a preliminary analysis on its evolution

Davy Landman[*], Alexander Serebrenik[*†], Jurgen Vinju[*†‡]
[*] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{Davy.Landman, Jurgen.Vinju}@cwi.nl
[†] Eindhoven University of Technology, Eindhoven, The Netherlands
a.serebrenik@tue.nl
[‡] INRIA Lille Nord Europe, Lille, France

I. Introduction

Is it useful to measure both Cyclomatic Complexity (CC) and Source Lines of Code (SLOC)? In previous work [1] we have analyzed the reported linear relationship between CC and SLOC. In our large corpus of Java projects, we could not find such a linear relationship. Raising questions for future work.

Object Oriented Programming (OOP) could cause the lack of a linear relationship between CC and SLOC. In an OO language, dynamic dispatch and polymorphism are used as an alternative to control flow statements. However, related work [2], [3] reported linear relationships for both C++ and Java.

As identified in our earlier work, there is an open question of the evolution of this relationship. Therefore we explorer a possible evolutionary argument: are the Java programs of today using more OOP? And does this cause the decreased power of SLOC to predict CC?

II. Research method

As a preliminary study of the evolution of this relationship, we select one software systems and calculate CC and SLOC for each method over the last 10 years. To summarize over this period, we sample only the methods in the system at the end of every full year (e.g. 2003–2013 range).

A. Hypothesis

Related work measured the linear relationship with Pearson's correlation (R^2). Similarly to our previous work [1], we will calculate both the Pearson and Spearman correlation. The Pearson correlation will be calculated before and after a power transform. Moreover, we will also perform the Breusch-Pagan test [4] to confirm non constant variance (heteroscedasticity).

We have formulated the following two hypothesis.

Hypothesis 1. *Older revisions of a software system have a stronger linear (Pearson) correlation between the CC and SLOC metrics for Java methods then newer revisions.*

Hypothesis 2. *Older revisions of a software system do have constant variance between the CC and SLOC metrics for Java methods.*

B. System

We have selected a single system out of the Qualitas Corpus, DrJava. It was selected due to its domain and age. It is a Java Integrated Development Environment (IDE) with over 3000 revisions since 2000. The system grew from 30 K SLOC in 2003 to 200 K SLOC in 2013. We chose an IDE since they contain elements of multiple domains.

C. Measuring SLOC and CC

We use the same tools as in our previous study [1]. Eclipse JDT is used to parse Java methods into Abstract Syntax Tree (AST) form. This AST is visited and for each node that would generate a fork in the Java control flow graph, 1 is added to the CC of that method. For SLOC we use Rascal to tokenize Jave into newlines, whitespace, comments and other words. These tokens are then used to calculate the SLOC of a method.

III. Results

A. Correlation

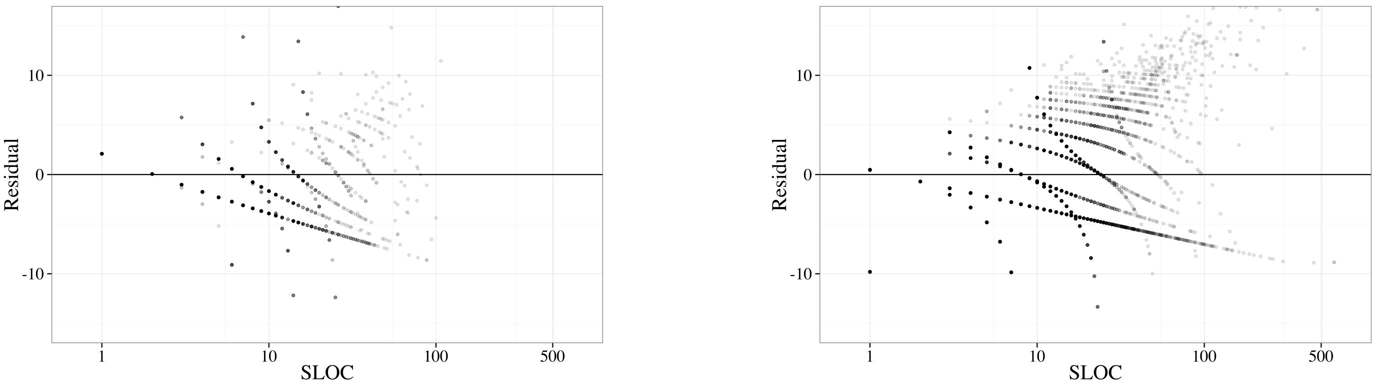
Table I contains the Pearson correlations before and after power transform, Spearman's correlation, and if the linear model was heteroskedastic.

TABLE I
Correlations between CC and SLOC for a period of 10 years and the total number of methods in that revision. All correlations have a high significance level ($p \leq 1 \times 10^{-16}$). Hetroskedasticy is checked by the Breusch-Pagan test (in all cases $p \leq 1 \times 10^{-16}$).

Year	Methods	R^2	$\log R^2$	ρ	Heteroscedastic
2003	3090	0.45	0.45	0.65	Yes
2004	4812	0.45	0.47	0.66	Yes
2005	9859	0.59	0.52	0.70	Yes
2006	10 262	0.56	0.47	0.67	Yes
2007	13 784	0.34	0.38	0.62	Yes
2008	14 998	0.35	0.39	0.62	Yes
2009	17 466	0.43	0.39	0.61	Yes
2010	19 765	0.44	0.40	0.61	Yes
2011	20 421	0.43	0.41	0.62	Yes
2012	20 470	0.42	0.42	0.63	Yes
2013	20 476	0.42	0.42	0.63	Yes

B. Scatter plots

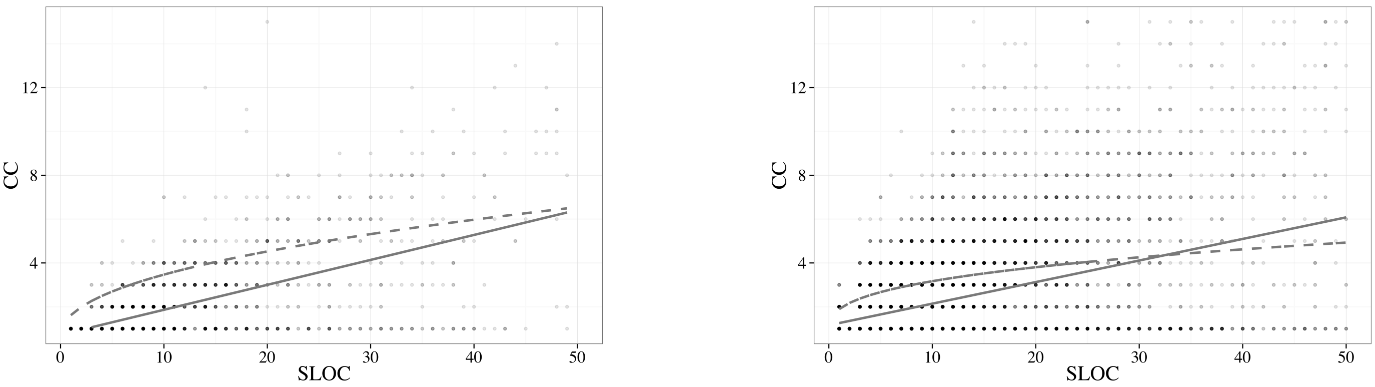
Figure 2 shows a zoomed-in ($\text{CC} \leq 20$ and $\text{SLOC} \leq 50$) scatter-plot of the methods of DrJava in 2003 and 2013. Due to



(a) 2003

(b) 2013

Fig. 1. Residual plot of the linear regression after the power transform, both axis are on a log scale. The non-constant variance complicates the interpretation of the linear regressions.



(a) 2003

(b) 2013

Fig. 2. Scatter plots of SLOC vs CC. The solid and dashed lines are the linear regression before and after the power transform.

the skewed-data, this figure still shows 98% of all data points. The two gray lines in the figure shows the linear regressions before and after the power transform. The gray scale gradient of the points in the scatter-plot visualizes how many methods have that combination of CC and SLOC: the darker, the more data points.

IV. Analysis

A. Hypothesis 1: correlation in older revisions

In Table I we see that although R^2 fluctuates over the years of DrJava's development, it remains near the 0.40, with the exception of 2005 and 2006. The increase in correlation could perhaps be explained by the big growth in 2005. However, we cannot confirm Hyptothesis 1, older versions of the software do not have a higher correlation.

B. Hypothesis 2: constant variance in older revisions

Table I shows that for all years the relation between CC and SLOC has non constant variance. The scatter plots in Figure 2 also visualize this growing variance, further shown in the residual plots in Figure 1. Therefore, we cannot confirm Hypothesis 2.

V. Discussion

We have presented a preliminary study on the evolution of the relationship between CC and SLOC. In the software system we analyzed, we did not observe a obvious change the linearity of the relationship. We also found that the heteroscedasticity reported in our previous work was present all versions of DrJava. Hetroscedasticity further complicates the inpretation of linear models.

In this abstract we have presented the evolution of one system. For future work, we would like to analyse the evolution of a whole corpus over the span of at least 10 years. Moreover, we are interested what other variables we might be measuring by comparing older version of the system with newer versions.

References

[1] D. Landman, A. Serebrenik, and J. Vinju, “Empirical analysis of the relationship between CC and SLOC in a large corpus of Java methods,” in *30th IEEE International Conference on Software Maintenance and Evolution, ICSME 2014*, 2014.

[2] K. E. Emam, S. Benlarbi, N. Goel, and S. N. Rai, “The confounding effect of class size on the validity of object-oriented metrics,” *IEEE Transactions on Software Engineering*, vol. 27, no. 7, pp. 630–650, 2001.

[3] G. Jay, J. E. Hale, R. K. Smith, D. P. Hale, N. A. Kraft, and C. Ward, “Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship,” *Journal of Software Engineering and Applications*, vol. 2, no. 3, pp. 137–143, 2009.

[4] T. Breusch and A. Pagan, “A simple test for heteroscedasticity and random coefficient variation,” *Econometrica*, vol. 47, no. 5, pp. 1287–1294, Sep. 1979.